Proceedings of the ASME 2023
International Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
IDETC/CIE2023
August 20-23, 2023, Boston, Massachusetts

DETC2023-115309

PREDICTION OF HUMAN REACHING POSE SEQUENCES IN HUMAN-ROBOT COLLABORATION

Jared Flowers
University of Florida
Gainesville, FL

Gloria Wiens
University of Florida
Gainesville, FL

ABSTRACT

In Human-Robot Collaboration (HRC), robots and humans must work together in shared, overlapping, workspaces to accomplish tasks. If human and robot motion can be coordinated, then collisions between robot and human can seamlessly be avoided without requiring either of them to stop work. A key part of this coordination is anticipating humans' future motion so robot motion can be adapted proactively. In this work, a generative neural network predicts a multi-step sequence of human poses for tabletop reaching motions. The multi-step sequence is mapped to a time-series based on a human speed versus motion distance model. The input to the network is the human's reaching target relative to current pelvis location combined with current human pose. A dataset was generated of human motions to reach various positions on or above the table in front of the human starting from a wide variety of initial human poses. After training the network, experiments showed that the predicted sequences generated by this method matched the actual recordings of human motion within an L2 joint error of 7.6 cm and L2 link roll-pitch-yaw error of 0.301 radians on average. This method predicts motion for an entire reach motion without suffering from the exponential propagation of prediction error that limits the horizon of prior works.

Keywords: Human-Robot Collaboration, Human Motion Prediction, Human-Robot Interaction

NOMENCLATURE

$h, \boldsymbol{H_p}, \boldsymbol{H_{act}}$	Single human pose, sequence of predicted human poses, and actual human pose sequence.
e_{w_t}	Euclidean error between the reaching wrist (left or right) and the target at time t in the sequence.
P_p, P_{tgt}	Human pelvis location and reaching target relative to P_p in the world coordinate frame.
$q_t, q_n, q_s, q_{uL}, q_{fL}, q_{uR}, q_{fR}$	Quaternions relating the world z-axis to the human torso, neck, shoulder-shoulder vector, left upper arm, left forearm, right upper arm, and right forearm, respectively.
$q_i = \langle w_i, x_i, y_i, z_i \rangle$	Quaternion i of the human, where w_i denotes rotation angle and x_i, y_i, z_i components to denote elements of rotation vector.
θ_i, v_i	Rotation angle and vector that determine i^{th} quaternion of the human.
P_{w_t}	Location of the reaching wrist (left or right) in the world frame at time t.
$D(i,j)$	Euclidian distance between the i^{th} point in one sequence and the j^{th} point in another sequence.
$W, w_k, w_{k(i,j)}$	Warp path that maps elements of one sequence to elements of another, an element of W which has an i and j component for each sequence, and either the i or j component of w_k.
z	Feature vector input to the neural network.
$a(c,i,j)$	Filter kernel input from node i,j in channel c from the previous neural net layer.
$o(c,i,j)$	Output of node i,j in a channel c in a layer of the neural network.
ϕ	Set of all learned neural network parameters.
v_{est},v_{act},S_{xy}	Wrist speed estimate, actual speed, and standardized residual error of the estimate.
$e_p(z_t, \phi_t)$	L1 error at time step t between the predicted pose sequence generated by the neural network and actual sequence from train/test samples.

1. INTRODUCTION

A challenge in human-robot collaboration (HRC) is coordinating human and robot motion. In HRC, humans and robots share a common workspace and work together in close proximity to accomplish tasks, e.g. in manufacturing. In an HRC

cell, the less coordinated human and robot motion is, occurrence of production delays and/or human discomfort becomes more likely. In the case of suboptimal coordination, the robot may have to stop and wait for the human to back away, causing production delay. The robot may also take trajectories that make the robot come close to the human, causing human discomfort and distrust in the robotic system. To improve human-robot coordination and avoid these problems, humans' trajectories must be predicted so robot motion can be adapted ahead of potential disruptions. In a manufacturing setting, the location of parts is known or easily determined, which provides the target for human reaching motions. Therefore, the prediction is the sequence of human poses generated from interpolations between the current pose and reaching target. This work presents a method for predicting a sequence of human poses based only on the human's current pose and the reaching target for the human's left or right wrist.

Human motion can be predicted at high and low levels within an HRC system. At the low level of prediction, a time sequence of human poses is predicted. At the high level, coarse human actions are classified, and the end point of human motion can be predicted, but without time dependence. Zhang et al. developed a recurrent neural network (RNN) architecture including units for independent human parts for predicting the end-point of human motion for a robot/human part handover [1]. Liu et al. used a Hidden Markov Model to generate a motion transition probability matrix for predicting next human coarse actions [2]. Maeda et al. proposed Probabilistic Motion Primitives to predict human intent and generate a corresponding robot motion primitive [3,4]. These methods can only provide the end-point for human motion at best. They do not provide details about the motion between start and end, limiting their potential to adapt robot motions to avoid predicted disruptions.

Previous works on predicting time sequences of human poses, meaning the motion between start and end, in a manufacturing domain have used filters and/or RNNs. Mainprice et al. fit Gaussian Mixture Models (GMMs) to many recordings of human reaching motions and predicted future motion with the GMMs [5]. Wang et al. used an autoregressive integrated moving average model applied to the elbow and wrist angles to predict human tabletop reaching motions [6]. Kanazawa et al. used Gaussian Mixture Regression (GMR) with Expectation Maximization to learn a model of human motion online [7]. Liu and Liu used an RNN to model human motion and used a modified Kalman filter to adapt RNN layer weights online [8]. Li et al. used a multi-step Gaussian Process Regression and previously recorded human trajectories to predict human reaching motion online [9]. Callen et al. developed a database of human motion models using Probabilistic Principal Component Analysis (PPCA) [10]. In many of these methods, the model predicts the next step based on the current step and then subsequent predictions are based on the previous prediction. Therefore, if there is a small error in predicting a relatively immediate step, that error will propagate through the prediction and result in exponential increase in error as the prediction horizon increases. Some of these works, such as GMMs and PPCA require a database of human trajectories which in turn requires computation time to compare current motion to a record.

Other state of the art human motion prediction methods have demonstrated good results for predicting human motion in general activities, such as walking and eating, represented in the Human3.6M dataset. Martinez et. al utilized a sequence-to-sequence architecture, which is an RNN with gated recurrent units (GRUs) that takes a sequence of recent poses and generates a predicted sequence of future poses [11]. Mao et al. encoded human pose trajectories using the Discrete Cosine Transform (DCT) and then used a Graph Convolutional Network which predicts future DCT coefficients based on a sequence of DCT coefficients [12]. Li et al. utilized a neural network consisting of encoders that use convolutional layers to generate hidden states based on long and short term input sequences and then use two fully connected layers to decode hidden states into pose sequences [13]. These methods generate predictions iteratively, causing exponential divergence of the prediction from the true trajectory over the time horizon. Therefore, they limit error analysis to predictions within a 1 second horizon. Such a short prediction horizon is infeasible for a manufacturing HRC setting where human motions are typically many seconds in duration.

The method herein uses a neural network to predict a sequence of human poses considering only the current human pose and reaching target as input. This method is designed to prevent the problem of error propagation over a long prediction horizon. The first step of this method is to warp the time scale of human motion observations in the training data to a dimensionless phase scale so each training sample shows consistent timing of changes in human pose elements. After conditioning the training data, a generative neural network is fit to the training data. The neural network assembled in this work is inspired by generator networks in Generative Adversarial Networks (GANs) [14,15]. Once the network is trained, it is used to predict a multi-step sequence of human poses. To use the prediction in the time domain, linear interpolation is used to match the multi-step prediction to a sequence having duration based on the anticipated human average speed.

The novelty of this work is development of a neural network and data pre/post conditioning to generate a predicted human pose sequence over a horizon of multiple seconds based only on the current human pose and relative reach target. This method utilizes the repetitiveness of human motion in manufacturing by considering the reaching target as an input. This method is unique in representing the human pose with quaternions so human link dimensions are preserved and the neural network inputs are continuous, enabling better network fit. Other representations are either not continuous or allow link lengths to change instantaneously. The method herein can also generate a prediction in real-time (faster than 30Hz) over a long horizon without suffering from exponential propagation of error that occurs with other works. This method is also trained and predicts based on data collected with a depth camera-based skeleton tracking system. Other methods utilize a more precise motion capture system for human tracking but require wearable sensing equipment, making them infeasible for a manufacturing setting.

The output of the method herein can be used as an input to proactive-n-reactive robot algorithms to avoid anticipated, time-varying delays. Fig. 1 shows how this human motion prediction method, shown by the large blue block, fits into the control scheme for a robotic system in an HRC workcell. The robot considers the task goal and a predicted time-sequence of human motion to plan robot motion that accomplishes the goal while avoiding the human. The robotic system then uses a safety controller to adjust robot speed or stop the robot along the planned path if the robot gets too close to the human. In the robotic system, real-time human pose is captured by the workcell sensor suite. A new human motion prediction can then be generated based on the human's reaching target (e.g. part to pick up) and current human pose. Targets can be determined by existing methods that locate objects based on image inputs [16]. The remaining sections of this paper are organized as follows: 2) methods, 3) results, and 4) conclusions. Section 2 is further divided into subsections: 2.1) human pose representation, 2.2) collection of training data, 2.3) preconditioning of the training data, 2.4) network architecture, and 2.5) postconditioning output to a time sequence of poses. Section 3 is divided into subsections: 3.1) training results, 3.2) prediction accuracy, and 3.3) implementation into an HRC system.

2. METHODS

The method in this work is composed of five parts. First, human pose is represented as pelvis cartesian location and a set of quaternions that relate each human link to the world z-axis. Second, a dataset is collected in which many iterations of various human motions are recorded as the human reaches to various target locations. Third, the recorded data are conditioned to have a consistent phase scale instead of a time scale to improve neural network training. Fourth, a neural network is created to predict a multi-step sequence of human poses the human will pass through to reach for a target, given the target and current human pose as input. Fifth, the network output is post processed to have a time scale that matches the estimated duration of human motion based on average motion velocities from the recorded data.

2.1 Human Representation

The human pose in this work is the stacked vector of the human pelvis location and the seven quaternions that define the axis for the torso, neck, shoulders, upper arms, and forearms:

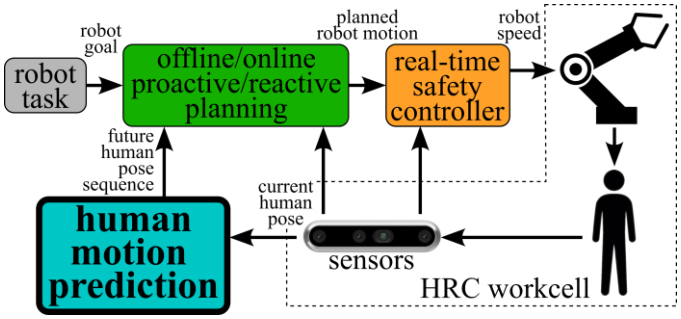

FIGURE 1: CONTROL BLOCK DIAGRAM FOR A ROBOTIC SYSTEM IN AN HRC WORKCELL.

$$h = [P_p, q_t, q_n, q_s, q_{uL}, q_{fL}, q_{uR}, q_{fR}]^T. \quad (1)$$

The cartesian pelvis location, P_p, is defined in the world coordinate frame. The $q_t, \ldots, q_{fR}$ are the human link quaternions, shown in Table 1 and fig. 2(A). Human link lengths and radii are required to fully define the volume each human link occupies. The method herein considers those parameters as constants determined a priori by the tracking system. The quaternions define a rotation about a vector to align the world z-axis with each human link's axis:

$$q_i = \langle w_i, x_i, y_i, z_i \rangle, \quad (2)$$

where the subscript i indicates one of the human link quaternions, as in (1) and Table 1. The quaternion elements are determined from a rotation angle (θ_i) and the unit vector (v_i) about which rotation is defined:

$$w_i = \cos\left(\frac{\theta_i}{2}\right), [x_i, y_i, z_i] = \sin\left(\frac{\theta_i}{2}\right) v_i. \quad (3)$$

The world frame z-axis would align with human link i if it were rotated by θ_i about v_i. Fig. 2(B) shows an example of the rotation angle and vector to align the world z-axis with the right forearm. Fig. 2(B) also shows the arm generalized to a cylinder for use by collision detection and path planning algorithms. Considering other possible angle/axis rotation representations, such as roll-pitch-yaw for example, the quaternion representation seems best suited for use with a neural network

Table 1. Links of the human kinematic chain.

Link	Description	Proximal Joint	Distal Joint	Link Quaternion
1	Torso/Spine	Pelvis	Spine	q_t
2	Neck	Spine	Shoulder MP	q_n
3	Shoulder- Shoulder	Shoulder MP	Shoulders	q_s
4	Left Upper Arm	Left Shoulder	Left Elbow	q_{uL}
5	Left Forearm	Left Elbow	Left Wrist	q_{fL}
6	Right Upper Arm	Right Shoulder	Right Elbow	q_{uR}
7	Right Forearm	Right Elbow	Right Wrist	q_{fR}

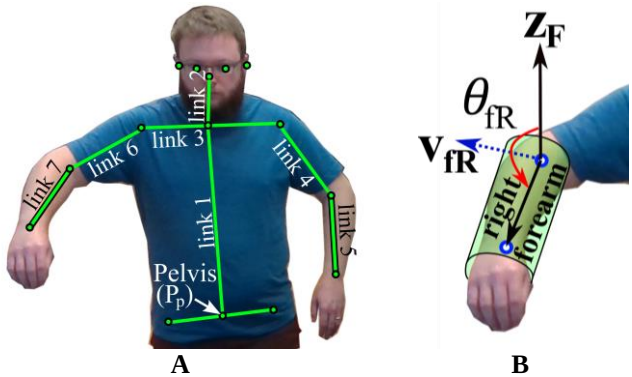

FIGURE 2: (A) LINKS OF THE HUMAN KINEMATIC CHAIN. (B) ROTATION ANGLE AND AXIS FOR THE RIGHT FOREARM QUATERNION AND CYLINDER LINK REPRESENTATION.

because quaternion elements are bounded between -1 and 1. The quaternion elements will also be continuous as the human moves. In contrast, the roll-pitch-yaw is either bounded but with discontinuity at $\pm\pi$ radians, or without discontinuity but allowing angles to tend toward $\pm\infty$. An example of this problem would occur if the human extended an arm outward and repeated full rotations of the arm about the shoulder. While it is physically impossible for a human joint to do a full revolution due to muscle and joint limits, sensing systems can perceive multiple full revolutions. Therefore, the quaternion representation is used to allow for the perception of full revolutions of human joints.

2.2 Collection of the Training Dataset

To amass a dataset of human motions, human pose was recorded while performing a variety of tabletop reaching tasks. Over 1,750 motion sequences were recorded per each arm. Each reaching motion had a target wrist cartesian location. An array of targets of known 3D positions was selected to cover the workspace in front of the human in the robotic cell, shown in fig. 3. Tabletop level targets are shown as green circles. The tips of rods extending upward from the table at each green circle by 15, 30, and 45 cm were used to create elevated targets, shown by blue rectangles. Human joint locations were tracked, using the two depth cameras circled in red in fig. 3, and converted into the quaternion representation via the method in [17]. The targets are in the range $x \in [-0.6,0.6]$, $y \in [0,0.6]$, $z \in [0,0.5]$ meters, with the human standing near the edge of the table near $[x, y, z] = [0,0.8,0]$m. For reference, the tabletop height is $z = 0$ meters. For each arm, motions included reaches to targets on both left and right sides of the pelvis to include some cross body motions. For reaches over long distances, motions could require the human to walk more than one step to reach the target. In this case, the prediction of motion may become significantly less accurate because the human has many more options for potential trajectories. Therefore, this work applies to human reaching motions in a tabletop setting in manufacturing where objects the human will interact with are within about one meter of the pelvis. For motions to objects more than one meter away, another algorithm such as in [18] could be used to generate relatively coarse predictions of occupancy at the expense of precision.

2.3 Conditioning of Training Data

The time required for humans to complete tasks likely varies over each iteration of the task, possibly due to distractions or tiring as work shifts progress. As humans reach for the same target over many iterations, the poses in the recorded time-series may be very similar, but will likely occur at different times through the motion. If the time scale of the training data were warped to have a consistent number of steps per sequence, then the effect of varying timing would be minimized, making each training record for a particular task as similar as possible. Trial and error showed that matching the time scale of all records to a common phase scale reduced the prediction error at the end of network training. Therefore, Dynamic Time Warping (DTW), specifically the FastDTW algorithm, was used to match all training records to a common phase scale [19].

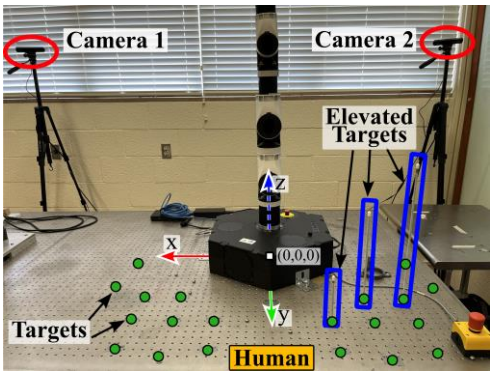

FIGURE 3: HRC WORKCELL USED IN COLLECTING DATA AND VALIDATION.

Consider two time-sequences of human poses, h_1 and h_2, both having the same time step. DTW outputs a mapping of time steps in h_1 to time steps in h_2, called a warp path which is denoted W. DTW uses dynamic programming to find the shortest warp path through a 2D grid where one dimension is the time step index of h_1, denoted i, and the other dimension is the time step index of h_2, denoted j. The path search starts from the first index of h_1 and first index of h_2 and must end at the last index of h_1 and last index of h_2. Dynamic programming iteratively determines a matrix the same size as the 2D h_1 by h_2 grid which indicates the distance of the shortest warp path to reach cell i, j from the start cell $(i = 0, j = 0)$. The distance matrix is updated iteratively according to:

$$D(i,j) = D(i,j) + \min[D(i-1,j), D(i,j-1), D(i-1,j-1)], \quad (4)$$

where $D(i,j)$ is the Euclidean distance between the i^{th} data point in h_1 and the j^{th} data point in h_2:

$$D(i,j) = \|h_1(i) - h_2(j)\|. \quad (5)$$

Dynamic programming iterations stop when the distance matrix reaches steady state values. Then a greedy search of that matrix finds the lowest cost warp path from start to end of h_1 and h_2:

$$W = \underset{\{w_1,\dots,w_K\}}{\operatorname{argmin}} \sum_{k=1}^{K} D(w_{ki}, w_{kj}), \quad (6)$$

where $w_k(i,j)$ is the k^{th} step of warp path W, indicating the mapping between i^{th} element of h_1 and j^{th} element of h_2. Therefore, w_{ki} indicates the i^{th} value of w_k and w_{kj} indicates the j^{th} value of w_k. FastDTW improves the speed of standard DTW by using a three-level approach for graph bisection to reduce the dimension of the search grid.

To precondition the input data, the Euclidean error between the reaching wrist and reaching target was determined across all time steps of each recorded motion sequence:

$$e_{w_t} = \|P_{w_t} - P_{tgt}\|, \; E_w = [e_{w_0}, \dots, e_{w_{t_{end}}}] \quad (7)$$

where P_{w_t} is the position of the reaching wrist, either left or right, and e_{w_t} is the wrist position error at timestep t and P_{tgt} is the

target wrist location. Then, per each recording, error was scaled so error is one at the start of motion and zero at the end of motion:

$$E_{w_{unit}} = \frac{E_w}{e_{w_{t_{start}}} - e_{w_{t_{end}}}}, \quad (8)$$

where $E_{w_{unit}}$ is the unitized error for the series of poses.

Plots of $E_{w_{unit}}$ versus time revealed most errors followed a decreasing sigmoid shaped curve. Therefore, a desired error curve was taken to be approximately the average of unitized error curves from all recorded motions, given by:

$$E_{desired} = 1 - \frac{1}{1+\exp(-10t+4.5)}, \; t \in [0,1]. \quad (9)$$

FastDTW was then used to determine the optimal mapping between the timesteps in the unitized error curve for each motion sample ($E_{w_{unit}}$) and the desired error curve ($E_{desired}$). Fig. 4 illustrates the warping of a sample's time scale by FastDTW. The orange dotted line is the desired error curve, blue dashed line is the observed error curve from the raw motion sample, and solid green line is the error curve after warping the time scale of the raw motion to match the desired curve. The black arrows show how FastDTW matches points in the raw error curve to points in the desired error curve. After the records' time scales were warped, then each record is down sampled to 10 points evenly spaced along the warped time scale, which will be called the phase scale herein. The phase scale indicates a percentage of reach motion completion at which a pose occurs. Trial and error showed that a sequence of 10 poses nearly matched the original sequence, but more poses did not improve accuracy. Now each training record presents a change in pose elements with more consistent scale than the raw samples.

2.4 Neural Network Architecture and Training

A neural network was assembled to predict a 10-step sequence of future human poses based on the current pose and human reaching target:

$$\boldsymbol{H_p} = f(z, \phi), \quad (10)$$

where $\boldsymbol{H_p}$ denotes the set of 10 sequential human poses for the prediction and ϕ are the network parameters. The prediction is

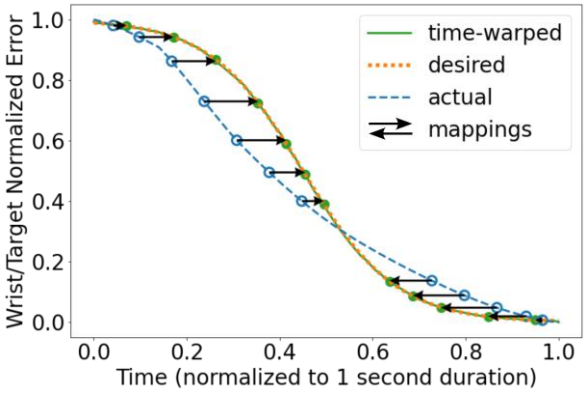

FIGURE 4: WARPING ACTUAL WRIST/TARGET ERROR TO THE DESIRED ERROR CURVE.

10 steps long because 10 poses nearly matched the actual sequence and increasing the number of poses didn't significantly improve error. The z is the vector input to the network given by:

$$z = [P_{tgt}, h]^T, \quad (11)$$

where P_{tgt} is the reaching target relative to the human's current pelvis location and h is the current human pose represented by pelvis location and quaternions. A separate neural network is used for predicting left arm reaches and right arm reaches. Trial and error led to the conclusion that separate networks reduced the difference between actual and predicted motions. The left arm network predicts pelvis location and quaternions for the torso, neck, shoulders, left upper arm, and left forearm. The right arm network predicts pelvis location and quaternions for the torso, neck, shoulders, right upper arm, and right forearm. When using either network, it is assumed that pose elements not predicted by the network (opposing upper arm and forearm quaternions) are held constant throughout the motion. Implementation of the left and right networks is discussed in subsection 3.3. Each neural network outputs a 10×23 matrix where each row is a phase step in the motion prediction. The columns correspond to the pelvis coordinates or quaternion elements in the human pose representation required for reaches with either left or right arm.

The network architecture is inspired by the generator network in a GAN [14,15]. The generator neural network in this method is a sequence of five transposed convolution layers as shown in Table 2 and fig. 5. In fig. 5, the input vector consisting of cartesian reaching target and current human pose is shown at the left. The blue blocks indicate the relative shape of the output of each network layer, with the text over the blocks indicating matrix size in the order: channels, height, width. The layer operation is indicated in the text below the blue blocks. The transposed convolution layers indicate the size of convolution kernel, kernel stride, and input padding in the format: height by width. The layer operations also indicate the activation function used by the layer, further explained below. A hyperbolic tangent activation function followed by network normalization of the human pose quaternions output is applied to the output of the final convolution layer. Each quaternion in the pose must have an L2 norm of 1 unit, so the human pose quaternions in each phase step of the network output are normalized.

Transposed convolutional layers generate output of larger height and/or width than that of the layer input. This property allows the network to generate a predicted pose sequence of higher dimension than the input vector. The transposed convolution layers convolve a filter kernel over the layer input to produce the layer output. The filter kernel is a matrix having height and width indicated in fig. 5. Filter kernel elements are learned dynamically by backpropagating network output error. The kernel is convolved with the input by shifting the kernel by the stride width across columns and stride height across rows and performing the sum of elementwise multiplication between the kernel and input at each kernel position. Each element of the transpose convolution layer output is determined by:

Table 2. Neural network layers.

Layer	Description	Input Size (c,h,w)	Kernel Size (h,w)	Stride (h,w)	Padding (h,w)	Output Size (c,h,w)
1	Conv. Transpose	26,1,1	(3,6)	(1,1)	None	256,3,6
2	Conv. Transpose	256,3,6	(3,3)	(1,1)	(1,1)	128,3,6
3	Conv. Transpose	128,3,6	(3,3)	(2,2)	(1,1)	64,5,11
4	Conv. Transpose	64,5,11	(3,3)	(2,2)	None	32,10,23
5	Conv. Transpose	32,10,23	(3,3)	(1,1)	(1,1)	1,10,23
6	Tanh, L2 quaternion norm	1,10,23	-	-	-	1,10,23

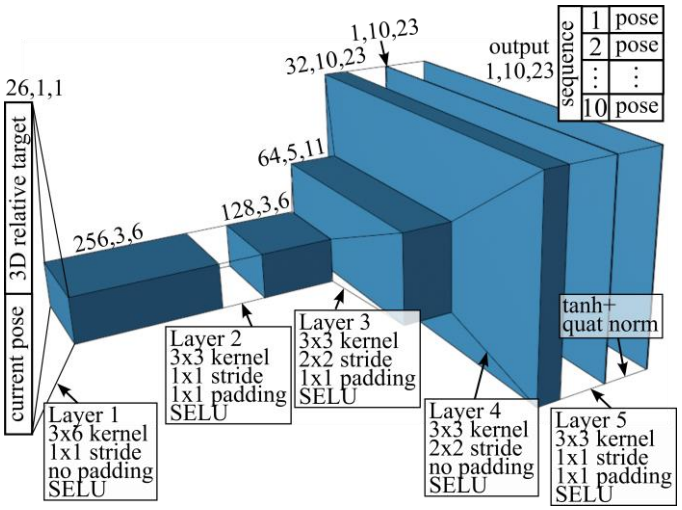

FIGURE 5: NEURAL NETWORK ARCHITECTURE FOR PREDICTING HUMAN POSE SEQUENCES.

$$o(c_o, n_i, n_j) = \sum_{m=0}^{c_i} \sum_{i=0}^{k_j} \sum_{j=0}^{k_j} k_{c_o}(i,j,m)\, a(m, n_i s_i + i, n_j s_j + j), \quad (12)$$

where $k_{c_o}(i,j,m)$ is the element at row i, column j, channel m of the kernel for output channel c_o [20]. The s_i and s_j indicate the stride width (rows) and height (columns), respectively. The n_i and n_j indicate the number of kernel strides that have occurred along the height and width of the input, respectively. The $a(m, n_i s_i + i, n_j s_j + j)$ indicates the layer input at row $n_i s_i + i$, column $n_j s_j + j$ for input channel m and c_i is the number of input channels. The kernel sizes, stride, and padding were selected to expand the network input into a matrix having shape 10×23 for output sequences with 10 phase steps. The number of channels in the output of each layer was found by trial and error. The number of channels in subsequent convolution layers were selected to be half that of the preceding convolution layer. Adding channels to the output of the first convolution layer reduced test set loss, but with diminishing returns over 256.

Batch normalization is applied after each convolution layer to improve the stability of network parameters while training [21]. The Scaled Exponential Linear Unit (SELU) is applied after each batch normalization operation [22]. The SELU activation function was selected to prevent exploding gradients and vanishing gradients by including a term for a positive gradient when its input is less than zero. Exploding gradients cause network parameters and outputs to tend toward infinite values. Vanishing gradients cause the gradients resulting from network output error to diminish as the error is backpropagated, so the gradient becomes too small to train layers near the input. Other activation functions, such as the Rectified Linear Unit (ReLU) and LeakyReLU, were also tested, but the SELU produced lower network loss at the end of network training.

When training the network weights, the prediction output by the network is compared to the actual sequences of future human poses from the conditioned training data. Since the network predicts a 10-step sequence of poses, 10 poses evenly spaced along the phase scale are taken from each training sample for comparison with the network output. The L1 (absolute) error between the predicted sequence ($\boldsymbol{H_p}$ after training iteration t) and actual sequence is used as the network loss function:

$$e_p(z_t, \phi_t) = \sum \left| \boldsymbol{H_p} - \boldsymbol{H_{act}} \right|, \quad (13)$$

where the summation is over all matrix elements; meaning across all channels, and height and width of each channel. The network parameters are adjusted based on the error using the Adam gradient-based optimizer and backpropagation [23]. For each network layer, the partial derivative of the "layer output" w.r.t. network parameters is determined from chain rule: product of the partial derivative of the "layer output" with respect to each SELU activation function and the partial derivative of the SELU activation function with respect to each network parameter. Backpropagation uses the partial derivates of layer outputs w.r.t. network weights to determine the effect the network loss should have on adjusting the network parameters using a form of gradient descent, such as the Adam optimizer.

2.5 Post Processing for Time-Series Prediction

The output of the neural network is the predicted sequence of human poses evenly spaced throughout a human task, but having a phase scale, not a time scale. Therefore, the 10-step prediction is interpolated to generate a prediction of a duration matching the anticipated duration of the human reaching motion. From the dataset of reaching motion human pose sequences, a relationship is observed in the scatter plot of the average speed during the reach motions versus distance between start and end wrist positions for the motion, shown as blue dots in fig. 6. Fig. 6 also shows the mean (μ, black dashed line) and the mean $\pm$ two standard deviations ($\pm 2\sigma$) (red line, shaded area) of the wrist speed as a function of reach distance. The line for mean wrist speed shows slight curvature, indicating a quadratic function will likely fit the speed (v_{est}) versus reach distance relationship better than a straight line. The quadratic best-fit line was:

$$v_{est} = -0.214d^2 + 0.659d - 0.0176, \quad (14)$$

where d is the distance in meters between start and end wrist positions. The best-fit line is shown as solid green in fig. 6. To ensure the best-fit line was not skewed by outliers, the modified Thompson-Tau method was used to omit outlier points. Speed

versus distance points whose standardized residual error were greater than two and whose neighbor points had residual error 0.5 less than it were rejected one at a time until no outliers met that criterion. This criterion is defined as:

$$\frac{|v_{est_i} - v_{act_i}|}{S_{xy}} > 2,\ S_{xy} = \sqrt{\frac{\sum_{i=1}^{n}(v_{est_i} - v_{act_i})^2}{n-2}}, \qquad (15)$$

where S_{xy} is the standardized residual error of the estimate, subscript i indicates the i^{th} data point, v_{act_i} is the observed speed, and n is the number of data points less previously rejected outliers. Since wrist speed can be predicted as v_{est} by (14), the estimated duration of the motion can be predicted according to:

$$t_{est} = \frac{\|P_w - P_{tgt}\|}{v_{est}} \qquad (16)$$

The number of time steps generated by interpolating the 10-step prediction is then $N = t_{est}/dt$, where dt is the desired prediction time resolution.

3. RESULTS

3.1 Network Training

To evaluate the success of training the neural network, the collected samples of pose sequences during reaching motions were randomly divided among a train set and a test set, having 80% and 20% of the total samples, respectively. The neural network was allowed 3000 epochs of training. In each epoch, the network parameters were adjusted based on the error in (13). During training, a batch size of 32 samples was used to reduce training time. The lower two curves (blue and red lines) in fig. 7 correspond to the sum of L1 error between the predicted and actual sequences in the train set averaged over each epoch for the left and right arm neural networks, respectively. At the end of each epoch, the network inferred a predicted pose sequence based on the start pose from each sample of the test set. The sum of L1 errors between the predicted and actual sequences of the test set are shown as the orange and green lines in fig. 7 for the left and right arm neural networks, respectively.

Fig. 7 shows that loss on the train set continues to decrease with diminishing returns after 3000 epochs, but loss on the test set stopped decreasing after about 450 epochs. This indicates the left and right networks overfit to the samples in the train set. The models can accurately predict a sequence that has been used for training but are much less accurate when predicting a sequence from the test set. Therefore, the error between actual and predicted pose elements was evaluated separately to determine if the test set loss could be attributed to a particular pose element.

Fig. 8 shows the L2 error between the actual and predicted pelvis location, averaged across all time steps of all samples in the train and test sets for each epoch for the network for left arm reaches. The orange line that plateaus at about 4 cm corresponds to the test set, the blue line that continues to decrease is from the train set, and the black dotted line is the average of the test set loss over the last 10 epochs. This figure shows the test set loss due to pelvis location reaches a minimum after about 350 epochs and then rises to a slightly higher steady state loss. The increase in loss after the minimum indicates that the pelvis location features contributed significantly to the plateau in test set loss.

Fig. 9 shows the L2 norm of the difference in predicted and actual quaternion elements for the torso, neck, left upper arm, and left forearm, averaged over all time steps of all test set samples for each epoch with the left reach network. Fig. 9 shows a moving average of train set losses and test set losses as dashed and solid lines, respectively, for each quaternion. To reiterate, fig. 8 and 9 are showing the network loss from fig. 7 broken out into individual pose elements. The test set losses for all pose quaternions in fig. 9 plateaued after more epochs than the pelvis location in fig. 8, with the exception of the neck quaternion (q_n). Fig. 9 even shows quaternion test set losses are still slightly decreasing after 3000 epochs. This means that the plateau in summary test set loss after 450 epochs shown in fig. 7 can be attributed to the challenge of learning pelvis displacements (fig. 8) and neck orientation (fig. 9) for reaching motions. Pelvis displacement prediction is an anticipated challenge considering the options a human has for reaching for a target relatively far away, such as one meter away. This type of reach could be accomplished by bending at the hips and extending an arm toward the target without moving one's feet, so the pelvis barely moves. Another option is to take a step towards the target and extend the arm so the torso can remain more upright, resulting in a pelvis displacement of many centimeters. Since reaches towards targets over about 0.7 meters away have these two

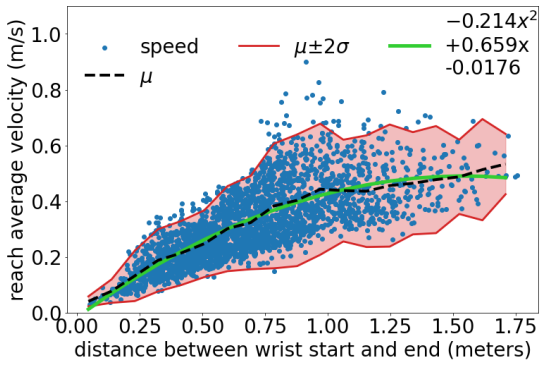

FIGURE 6: AVERAGE WRIST SPEED DURING REACH MOTION VERSUS EUCLIDEAN DISTANCE BETWEEN WRIST START AND END POSITIONS.

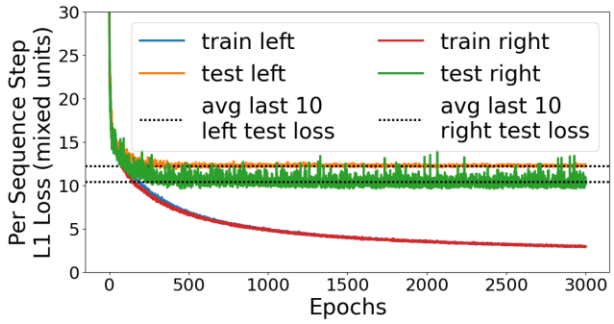

FIGURE 7: NETWORK OUTPUT LOSS ON THE TRAIN AND TEST SETS OVER EPOCHS OF TRAINING.

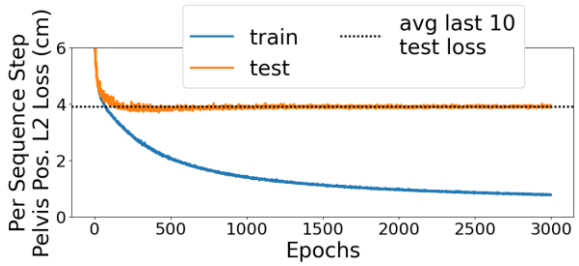

FIGURE 8: L2 ERROR BETWEEN PREDICTED PELVIS AND ACTUAL PELVIS LOCATION ON THE TRAIN AND TEST SETS.

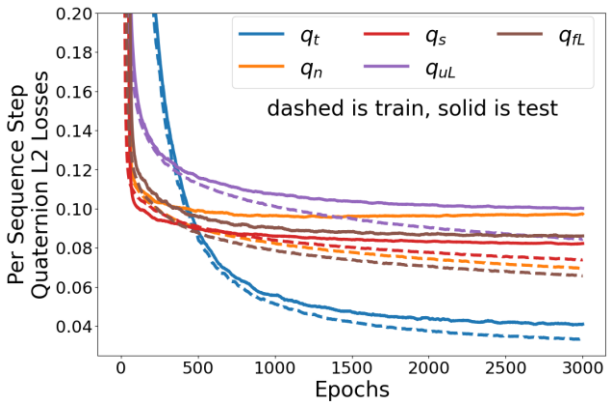

FIGURE 9: L2 ERROR BETWEEN PREDICTED QUATERNION ELEMENTS AND ACTUAL QUATERNION ELEMENTS, PER HUMAN LINK, ON THE TRAIN AND TEST SETS.

options, the network less accurately predicts pelvis displacement. Neck orientation prediction is also a challenge since the human doesn't always look at the reach target. Plots of pelvis displacement prediction error and quaternion element prediction error such as fig. 8 and fig. 9 for the right arm are nearly same as for the left arm, resulting in the same conclusion about the plateau in network loss on the test set.

3.2 Predication Accuracy

The goal of this work was development of a method of predicting human motion sequences. Therefore, the accuracy of the predicted sequence relative to actual motion sequences is a primary concern. One metric for assessing prediction accuracy is the Euclidean error between the predicted reaching wrist location and the target at the end of each motion, denoted reach position error herein. Table 3 shows the reach position error averaged over samples from the test set for the left and right networks and combined results for the final step of the sensed and predicted sequences. The 2nd and 3rd columns show error in sensed and predicted wrist position, each relative to the true reach target described in subsection 2.2. The 4th column shows error between sensed and predicted position. The 80% trimmed mean was used to exclude data in the lower and upper 10% which may be outliers due to error in sensing. These statistics show 10.6 cm error between the final wrist position in predicted sequences and the reach target. The recorded data also show average sensor wrist measurement error of 10.2 cm from the tracking system. Since the average sensor wrist errors from the recorded dataset are nearly as large as the error of the predicted motion, inaccuracy of the sensing system used to generate the datasets contributes significantly to inaccuracy in predictions generated by the networks. As mentioned earlier, the difficulty in predicting pelvis displacement is another source of prediction error. Fig. 10 shows an example of an actual and predicted human pose sequence. The lower five subplots show the prediction of quaternion elements closely matches the actual. The top subplot shows pelvis displacement, indicating a prediction error of about 3cm in the x direction at the end of the sequence.

Prediction error along entire motion sequences, not just the final sequence step, also provides insight into the usefulness of this method for motion prediction. Table 4(A) shows errors between the predicted and actual pose, averaged over all human joints/links, all time steps, and all sequences in the test set, considering the post-processed network output as the predicted and the raw test set samples as the actual, meaning predicted and actual both use a time scale and not a phase scale. The second column shows the Euclidean norm joint errors averaged over all human joints. The third column shows the Euclidean norm between quaternion elements per human link quaternion, and fourth column shows Euclidean norm between link roll-pitch-

Table 3. Trimmed mean (80%) position error between wrist and reach target at the end of reach motions, averaged over all samples.

Side	Sensor (camera) Wrist Error (cm)	Predicted Wrist Error (cm)	Predicted/Sensed Difference (cm)
Left	9.2	9.8	7.7
Right	11.3	11.4	8.3
Combined	10.2	10.6	8.0

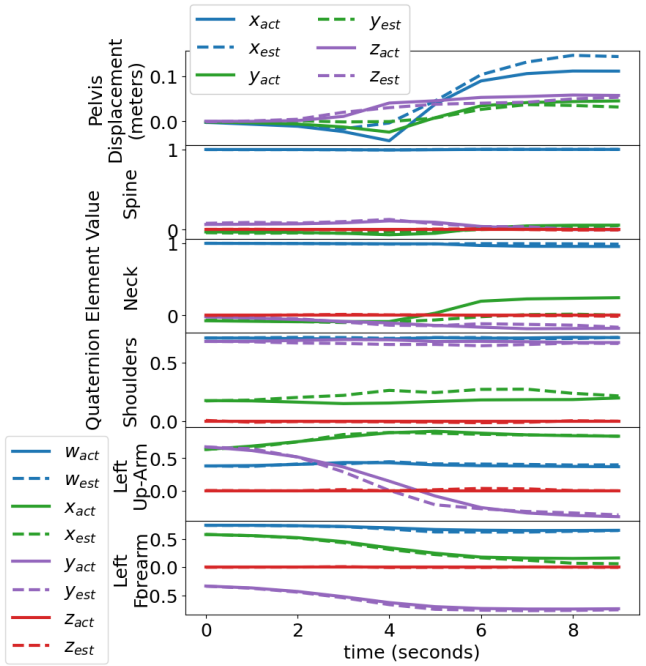

FIGURE 10: PREDICTED AND ACTUAL SEQUENCES FOR THE HUMAN POSE ELEMENTS FOR A LEFT ARM REACH.

yaw, averaged over all human links. Table 4(B) shows the error comparison between the network output without post-processing and the test set samples after pre-processing, meaning the predicted and actual sequences both use the phase scale instead of a time scale. The average error between predicted and actual joint locations averaged over all joint locations and all sequences in the test set was 7.6 cm with the time scale and 5.8 cm with the phase scale. The error in Euclidean norm of roll-pitch-yaw averaged over all links, time steps, and test set sequences was 0.301 radians when using the time scale and 0.198 radians with the phase scale. Smaller phase scale errors than time scale errors indicate that timing of motions makes the poses less predictable. However, the time scale results are more realistic since the method output is a time-series of predicted human poses. Table 4 also shows error in the L2 norm of quaternion differences, but it is less interpretable than roll-pitch-yaw since quaternion elements have mixed units.

While the position and orientation errors are larger than desired, they are comparable to the result in a recent work on human motion prediction for activities in the Human3.6M dataset over a one second horizon [12]. The method herein provides a significant advantage in that generated predictions are over a considerably longer time horizon than other recent works. Prior works predict the next iteration based on prediction at the current iteration. This causes prediction error to increase exponentially as the prediction horizon increases, so prior works don't report prediction error beyond a one second horizon. The method herein predicts over the entire duration required for a human to reach for a target location, which was up to 3.5 seconds for recorded motions.

3.3 Implementation into HRC System

When the method herein is part of a larger HRC system, such as in fig. 1, it can be triggered to generate a prediction once a reaching target is received. This method can also predict continuously while motions are in progress. Since there is a separate network to predict left and right arm motions, this method must be accompanied by an algorithm that predicts which arm will perform the reach motion. One possibility is to assume that if the reach target is to the right of the pelvis, then predict motion for the right arm, otherwise predict for the left arm. If the person is currently holding a piece with one arm, then it can be assumed that the reach will be performed with that arm. An advantage of the method herein is prediction inference in less than 2 milliseconds. This was the performance on an Intel i9 computer with an Nvidia RTX3070 GPU, using the GPU for model inference and CPU for pre/post processing of data. This means that if the actual human motion starts to deviate from the prediction while reaching for a target, this method can rapidly predict a new, more accurate prediction. Deviations requiring rapid updates could include changes in poses, timing, right/left arm usage, or change of target.

Table 4. Trimmed mean (80%) of L2 norm of difference between predicted and actual joint locations, quaternions, and roll-pitch-yaw over all human links and all time steps of all test set sequences, with two comparisons, A and B.

A. Post-processed network output vs. raw test set samples			
Side	Avg. Joint Error (cm)	Avg. Quaternion Error	Avg. RPY Error (radians)
Left	8.3	0.113	0.325
Right	7.6	0.105	0.274
Combined	7.6	0.105	0.301
B. Raw network output vs. time-warped test set samples			
Left	6.3	0.074	0.205
Right	5.8	0.071	0.190
Combined	5.8	0.071	0.198

Fig. 11(B-E) shows a more realistic scenario of moving a piston-rod assembly from a fixture on the table to an engine block in chronological order. It shows an initial, final, and two intermediate actual and predicted human poses. The human is represented as the blue and orange collections of cylinders corresponding to actual and prediction, respectively. The engine parts are shown as small and large green boxes corresponding to a piston-rod assembly and engine block, respectively. Fig. 11(A) shows the engine block in the lower part of the image and piston-rod assembly in the upper right corner, from the human's perspective. The reaching target for the right arm is shown as the red sphere over the engine block. Fig. 11(A) also shows the right wrist trajectory as a dashed blue arrow. Fig. 11 shows the method herein predicted a sequence that closely matched both the pose and timing of the actual sequence.

4. CONCLUSION

In summary, this work developed a neural network and pre/post processing to predict sequences of human poses for reaching motions in an HRC workcell. The input to the method is the human's current pose and the reaching target for either the left or right wrist. A key advantage of this method is ability to predict motion over the entire duration of a reach motion, up to 3.5 seconds in train/test set motions. The method herein does not suffer from exponential propagation of errors experienced by

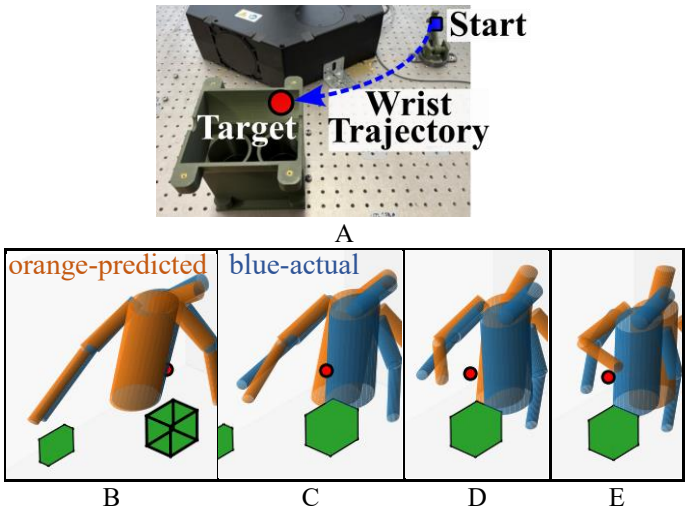

FIGURE 11: (A) ENGINE BLOCK AND PISTON-ROD ASSEMBLY. (B-E) FOUR TIME STEPS FROM A PREDICTED AND ACTUAL MOTION SEQUENCE IN INCREASING TIME FROM IMAGE B TO E.

prior works, limiting them to a horizon of 1 second. The method herein provides predicted human pose sequences for input to other robot control algorithms to permit proactive avoidance of humans. Proactive responses can mitigate production delays and reduce human discomfort in HRC. Future work will incorporate this method with proactive robot path planning and seek ways to reduce prediction error, such as improved sensing technology.

ACKNOWLEDGEMENTS

Funding was provided by the NSF/NRI: INT: COLLAB: Manufacturing USA: Intelligent Human-Robot Collaboration for Smart Factory (Award I.D. #:1830383). Any opinions, findings and conclusions or recommendations expressed are those of the researchers and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Zhang, J., Liu, H., Chang, Q., Wang, L., Gao, R. X, 2020. "Recurrent Neural Network for Motion Trajectory Prediction in Human-Robot Collaborative Assembly." *Cirp Annals-Manufacturing Technology*. 69(1): 9–12. DOI 10.1016/j.cirp.2018.04.066.

[2] Liu, H., and Wang, L. 2017. "Human Motion Prediction for Human-Robot Collaboration." *J. Manuf. Sys* 44: 287–94. DOI:10.1016/J.JMSY.2017.04.009.

[3] Maeda, G.J., Ewerton, M., Lioutikov, R.,Amor, H.B., Peters, J., and Neumann, G. "Learning Interaction for Collaborative Tasks with Probabilistic Movement Primitives." Proc. of the IEEE-RAS ICHR, 2014, pp. 527-534. DOI:10.1109/HUMANOIDS.2014.7041413.

[4] Maeda, G.J. Neumann, G., Ewerton, M., Lioutikov, R., Kroemer, O., and Peters, J. 2017. "Probabilistic Movement Primitives For Coordination Of Multiple Human–Robot Collaborative Tasks." *Autonomous Robots*. 41: 593-612. DOI:10.1007/s10514-016-9556-2.

[5] Mainprice, J., and Berenson, D. 2013. "Human-Robot Collaborative Manipulation Planning Using Early Prediction of Human Motion." In Proc. of the IEEE/RSJ IROS, 2013, pp. 299–306. DOI:10.1109/IROS.2013.6696368.

[6] Wang, Y., Sheng, Y., Wang, J., and Zhang, W. 2018, "Optimal Collision-Free Robot Trajectory Generation Based on Time Series Prediction of Human Motion," *IEEE Robot. Autom. Lett.* 3(1): 226-233, DOI: 10.1109/LRA.2017.2737486.

[7] Kanazawa, A., Kinugawa J., and Kosuge, K. 2019. "Adaptive Motion Planning for a Collaborative Robot Based on Prediction Uncertainty to Enhance Human Safety and Work Efficiency," *IEEE Trans. Robot.* 35(4): 817-832, DOI: 10.1109/TRO.2019.2911800.

[8] Liu, R., and Liu, C. "Human Motion Prediction Using Adaptable Recurrent Neural Networks and Inverse Kinematics." In Proc. of the IEEE Am. Control Conf., 2021, pp. 3725-3730. DOI: 10.23919/ACC50511.2021.9482655.

[9] Li, Q., Zhang, Z., You, Y., Mu, Y., and Feng, C. 2020. "Data Driven Models for Human Motion Prsediction in Human-Robot Collaboration". *IEEE Access* 8: 227690-227702. DOI: 10.1109/ACCESS.2020.3045994.

[10] Callens, T., Van der Have, T., Van Rossom, S., De Schutter, J., and Aertbeliën, E. 2020. "A Framework for Recognition and Prediction of Human Motions in Human-Robot Collaboration Using Probabilistic Motion Models." *IEEE Rob. Autom. Lett.* 5(4): 5151-5158. DOI 10.1109/LRA.2020.3005892.

[11] Martinez, J., Black, M.J., and Romero, J. "On Human Motion Prediction Using Recurrent Neural Networks." In Proc. of the IEEE CVPR, 2017, pp. 4674-4683. DOI: 10.1109/CVPR.2017.497.

[12] Wei, M., Liu, M., Salzmann, M., and Li, H. "Learning Trajectory Dependencies for Human Motion Prediction." In Proc. of the IEEE/CVF ICCV, 2019, pp. 9488-9496. DOI: 10.1109/ICCV.2019.00958.

[13] Li, C., Zhang, Z., Lee, W.S., and Lee, G.H.. "Convolutional Sequence to Sequence Model for Human Dynamics." In Proc. of the IEEE/CVF CVPR, 2018, pp. 5226-5234. DOI: 10.1109/CVPR.2018.00548.

[14] Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., and Bengio, Y. 2014. "Generative Adversarial Nets." *Adv. Neural Inf. Process. Syst*. 27. DOI: 10.48550/arXiv.1406.2661.

[15] Radford, A., Metz, L., and Chintala, S. 2015. "Unsupervised Representation Learning With Deep Convolutional Generative Adversarial Networks." *CoRR*. DOI: 10.48550/arXiv.1511.06434.

[16] Papadaki, A.I., and Pateraki, M. 2023. "6D Object Localization in Car-Assembly Industrial Environment." *Journal of Imaging* 9(3): 72. DOI: 10.3390/jimaging9030072.

[17] Flowers, J., and Wiens, G. "Comparison of Human Skeleton Trackers Paired with a Novel Skeleton Fusion Algorithm," Proc. of the ASME MSEC, 2022, pp. 10. DOI: 10.1115/MSEC2022-85269

[18] Pellegrinelli, S., Moro, F.L., Pedrocchi, N., Tosatti, L.M., and Tolio, T.A.M. 2016. "A Probabilistic Approach to Workspace Sharing for Human–robot Cooperation in Assembly Tasks." *Cirp Annals-Manufacturing Technology* 65(1): 57–60. DOI:10.1016/J.CIRP.2016.04.035.

[19] Salvador, S., and Chan, P. 2007. "FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space." *Intell. Data Anal.* 11(5): pp. 561-580. DOI: 10.5555/1367985.1367993.

[20] Dumoulin, V., and Visin, F. 2016. "A Guide to Convolution Arithmetic for Deep Learning." *ArXiv*. DOI: 10.48550/arXiv.1603.07285.

[21] Ioffe, S., and Szegedy, C. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In Proc. of the ICML, 2015, 37: pp. 448-456. DOI: 10.48550/arXiv.1502.03167.

[22] Klambauer, C., Unterthiner, T., Mayr, A., and Hochreiter, S. 2017. "Self-Normalizing Neural Networks." *Adv. Neural Inf. Process. Syst*. 30. DOI: 10.48550/arXiv.1706.02515.

[23] Kingma, D.P., and Ba, J. "Adam: A Method for Stochastic Optimization." In Proc. of the ICLR. 2015. DOI: 10.48550/arXiv.1412.6980.

